Unsupervised Relation Detection using Automatic Alignment of Query Patterns Extracted from Knowledge Graphs and Query Click Logs

Panupong Pasupat\textsuperscript{1}; Dilek Hakkani-Tür\textsuperscript{2}

\textsuperscript{1}Stanford University, Palo Alto, CA, United States
\textsuperscript{2}Microsoft Research, Mountain View, CA, United States

ppasupat@cs.stanford.edu, dilek.hakkani-tur@microsoft.com

Abstract

Traditional methods for building spoken language understanding systems require manual rules or annotated data, which are expensive. In this work, we present an unsupervised method for bootstrapping a relation classifier, which identifies the knowledge graph relations present in an input query. Unlike existing work, we utilize only one knowledge graph entity instead of two for mining relevant query patterns from query click logs. As a result, the mined patterns can be used to infer both explicit relations (where the objects of the relations are expressed in the queries) and implicit relations (where the objects of the relations are being asked about). Using only the mined queries, the final classifier achieves an F-measure of 55.5\%, which is significantly higher than the previous unsupervised learning baselines.

Index Terms: conversational understanding systems, relation detection, search query click logs, unsupervised learning, semantic graph

1. Introduction

In a dialog system, the spoken language understanding (SLU) module receives transcribed speech queries and extracts their semantic information, which can be used for decision making and response generation [1, 2]. We focus on building a relation detector, which outputs all relations expressed in the query (e.g., “Who played Jake Sully in Avatar” has relations acted by, character name, and movie name). These relations are used to form queries to databases or knowledge graphs in order to generate an appropriate response [3].

Most approaches for building SLU systems depend on either complex hand-crafted rules, which require time and expertise to write, or supervised learning, which requires a large amount of human-annotated data. However, designers of SLU systems may reduce supervision by using external resources that relate natural language to some computable semantic structures. We focus on two resources: knowledge graphs, which represent relations between entities as large directed graphs, and web search query click logs, which link search queries to the URLs that the users click on.

In this work, we propose a totally unsupervised method for bootstrapping a relation detector. Instead of using manually annotated sentences, we acquire natural language queries in the domain of interest from search query click logs. Subsequently, we use knowledge graphs and URLs from the click logs to filter the queries and automatically label them with relations. Our previous work on relation detection [3] uses a similar distant supervision approach that automatically annotates sentences with the intended relations when both of the related entities appear in the search queries or web documents. However, such an approach only targets explicit relations, as it requires the subjects and objects of the relations to both be specified in the queries (e.g., the director name relation in “Find Avatar movie directed by James Cameron”), and hence makes limited use of related search queries. With our new methods, we can also infer implicit relations, where the values of the relations are being asked about and thus left unspecified (e.g., the directed by relation in “Who made Avatar”). Explicit and implicit relations are inferred with separate but related techniques: for explicit relations, we use knowledge graph entity-relation-entity triples to automatically label the objects of the relations, while for implicit relations, we consider the entities that can invoke the relations and leverage the query patterns mined from those entities.

The final classifier which uses only the mined data achieves a micro F-measure of 55.5\%, a large improvement over previous unsupervised learning baselines.

The rest of the paper is organized as follows. The next section formally introduces the task, knowledge graphs, and query click logs. Section 3 presents the relevant work that uses the same external resources. In Section 4 we describe our approach, the results of which are presented in Section 5. Finally, Section 6 concludes the paper.

2. Relation detection using knowledge graphs and query click logs

In this section, we describe the relation detection task and the resources we use to tackle the task: large-scale semantic knowledge graphs (e.g., Freebase [4]) and search query click logs.

2.1. Knowledge graphs

We focus on the relations that are present in graphical knowledge bases, or knowledge graphs for short. A knowledge graph, as illustrated in Figure 1, is a directed graph where each node represents an entity (e.g., Avatar or James Cameron) and each labeled edge represents a relation between two entities (e.g., directed by). Each entity belongs to one or more types (e.g., Avatar belongs to the film type), and each type has a schema specifying which relations should originate from the entities of that type (e.g., an entity of type film will have a directed by relation to an entity of type film director).

2.2. Relation detection task

We consider the relation detection task: given a transcribed natural language query, we want to determine all relations expressed in the query [3]. For example, both “Show me movies..."
by James Cameron” and “Who directed Avatar” contain the relation directed by, but only the first query contains the relation director name. These relations can be regarded as building blocks toward full language understanding, since more complex representations of the query, such as SPARQL knowledge graph queries or semantic logical forms, will contain these relations.

2.3. Query click logs

To obtain natural language queries in an unsupervised fashion, we use query click logs (QCLs), which record the URL that each user chose in a search engine after issuing a query. Along the line of [5], we represent QCLs as a weighted undirected bipartite graph: the queries and the URLs form two sides of nodes, and the weight of the edge between a query node and a URL node indicates the number of users who issued the query and then clicked on the URL.

3. Related work

Earlier work on statistical spoken language understanding (SLU) employs supervised learning, typically treating intent detection as multi-class classification and slot filling as sequence labeling [6, 7]. The training data for these tasks is annotated according to a task-specific semantic representation [8]. To decrease the cost of acquiring and labeling training data, a group of studies has investigated methods to adapt generic semantic representations, associated annotated data, and clustering methods for training task-independent models [9, 10, 11].

Another research trend is the application of pre-existing structured data in language understanding. In particular, with the emergence of large knowledge bases (Freebase, Yago2, DBPedia, Satori), many systems rely on knowledge graphs for distant supervision. For example, given large text corpora (e.g., Wikipedia or ClueWeb), information extraction systems can find sentences containing pairs of entities with some target knowledge graph relation, and then use the extracted sentences to train a high-precision relation extractor [12, 13, 14]. Instead of large corpora, the Web can also be used to supply relevant sentences by scraping the search engine snippets when the pairs of entities are used as search queries [3]. The alignments between the sentences (surface forms) and the underlying knowledge graph relations can also be utilized in downstream tasks such as classifying relations for answering factoid questions [15, 16, 17].

Besides knowledge graphs, query click logs have also been used to build SLU systems in an unsupervised fashion. QCLs were originally used to improve search results or suggest similar search queries [18, 19, 20], but as QCLs contain a large amount of text (search queries) with noisy semantic annotation (URLs), they are also used in many language understanding tasks such as acquiring related entities [21], clustering entities into semantic classes [22, 23], user intents detection [24], entity type classification [25], and knowledge acquisition [26]. In our previous work, the queries mined from QCLs are used as examples to classify query domains [27] and detect new query intents [28]. This work extends from the previous work by (1) using the relation model, which encompasses both intents and slots, (2) proposing methods for mining queries that require only one pivot entity instead of two, and (3) using aggregate pattern statistics of the queries in a novel way to detect both explicit and implicit relations.

4. Approach

We now describe our approach to unsupervised relation detection, which includes three major steps. First, we mine the queries related to the entities of interest from the query click logs (Section 4.1). Then, we use the queries to infer two types of relations, explicit (Section 4.2) and implicit (Section 4.3). Finally, the queries with inferred relations are used to train a combined relation classifier (Section 4.4).

4.1. Mining entities and queries

The first step of the pipeline is finding knowledge graph entities in the domain of interest. For example, in the movie domain, we want to find the list of all movies as well as their attributes (e.g., directors, actors, characters). We start by listing all entities of the central type corresponding to the desired dialog domain (e.g., Freebase film type for the movie domain). Then, for each entity $e_c$ of the central type (e.g., from Figure 1, $e_c = Avatar$), we compute the property list $P(e_c)$ of entities that are related to $e_c$. Formally, $P(e_c)$ includes:

- entities $e$ with an incoming relation from $e_c$ (e.g., $e = James Cameron$ via the relation directed by)
- entities $e$ reachable from $e_c$ within two relations via a mediator node (e.g., $e = Jake Sully$ via the relations starring and character)
- $e = e_c$ itself (e.g., $e = Avatar$)

Figure 1: A small portion of the knowledge graph. The gray circle is a mediator node representing an (actor, character) pair.

Figure 2: Query mining. From a property entity $e$, we find the corresponding URLs from either the seed queries (left) or the knowledge graphs (right), and then mine the corresponding search queries from the QCL.
from either stop words or
queries. In this work, we use a simpler technique derived
turn out to be “keyword” queries composed of noun phrases
ation format, which is stylistically similar to spoken language
an advantage that a large amount of search queries are in ques-
tions. For each entity e, we use approximate string

directed by
the implicit relation
character
name
movie name
(200,000 automatically labeled examples in our ex-
periments. To balance the relation labels, we impose that half
of the examples have the movie name relation while the other
examples do not.

4.3. Inferring implicit relations

Implicit relations are the relations whose objects are being
asked about and thus are left unspecified. For example, the
query “Who directed Avatar” has the implicit relation directed
by because it asks about the unspecified director’s name.

To infer implicit relations, we use a property of the QCL
illustrated by the following example. Consider queries of the
form “Who directed [movie name]”. Most of the time, users
who enter such queries will click on the official or encyclo-
dic pages about the movie; however, occasionally some users will
click on web pages about the director of the movie. In this case,
we may infer that the query pattern “who directed …” has the
implicit relation directed by. More generally, if the entity e
appearing in the example does not appear in the query
q, we may infer that the entity is likely the (missing) object
of an implicit relation in the query.

Using the intuition above, we create a dataset D_I for
training an implicit relation classifier as illustrated in Figure 4.
Consider an entity e ∈ P(e_c) and a query q mined for e. If the name
of e does not appear in q, then we translate the path p from e_c
in q to e into an implicit relation (e.g., p = directed by) in the example
above translates to the directed by relation). To reduce noise,
we filter out some out-of-domain queries by removing queries
q that do not contain any entity related to e.

The implicit relation dataset D_I contains 340,000 examples.
In addition to creating the dataset, we can also derive a
list of generic query patterns for each implicit relation by col-
lapsing entities e′ ∈ P(e) that appear in the queries into place-
holders based on the path between e and e′ (e.g., “Who directed
Avatar” becomes “who directed [film]”). Table I shows several
examples of good query patterns.

4.4. Combined classifier

After we obtain training data D_E for explicit relations and D_I
for implicit relations, we train a combined relation classifier
To evaluate our approach, we use the movie domain relation dataset from [32]. The dataset contains 3,338 training and 1,084 test examples. However, except for the supervised learning upper bound, we ignore the training examples and train the classifiers from the mined queries instead.

As mentioned earlier, we treat relation detection as a multi-class, multi-label classification problem. Each dataset example contains a transcribed query and one or more relations extracted from the annotated SPARQL database query. The dataset contains a total of 70 relations in total. The average number of relations per query is 2.58, and 9.5% of the queries are marked as ambiguous.

### 5.1. Dataset

To evaluate our approach, we use the movie domain relation dataset from [32]. The dataset contains 3,338 training and 1,084 test examples. However, except for the supervised learning upper bound, we ignore the training examples and train the classifiers from the mined queries instead.

As mentioned earlier, we treat relation detection as a multi-class, multi-label classification problem. Each dataset example contains a transcribed query and one or more relations extracted from the annotated SPARQL database query. The dataset contains a total of 70 relations in total. The average number of relations per query is 2.58, and 9.5% of the queries are marked as ambiguous. The original paper includes more information about the specifics of the dataset [32].

### 5.2. Results and discussions

Table 2 shows the micro F-measure of our experiments. Unless stated otherwise, we train classifiers using isctiboost [34], an adaptive boosting framework, with 10,000 iterations. We use common features for text classification including n-gram features (n = 1, 2, 3) and weighted gazetteer features calculated using the populated semantic graph [32].

<table>
<thead>
<tr>
<th>URL source</th>
<th>Classifier</th>
<th>n-grams</th>
<th>n-grams + w-gaz</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Majority</td>
<td>27.6</td>
<td>43.3*</td>
</tr>
<tr>
<td>seed</td>
<td>Explicit (D_E)</td>
<td>34.0</td>
<td>36.2</td>
</tr>
<tr>
<td>seed</td>
<td>Implicit (D_I)</td>
<td>16.8</td>
<td>17.0</td>
</tr>
<tr>
<td>seed</td>
<td>Combined</td>
<td>43.1</td>
<td>43.0</td>
</tr>
<tr>
<td>KG</td>
<td>Explicit (D_E)</td>
<td>39.6</td>
<td>42.7</td>
</tr>
<tr>
<td>KG</td>
<td>Implicit (D_I)</td>
<td>29.3</td>
<td>29.3</td>
</tr>
<tr>
<td>KG</td>
<td>Combined</td>
<td>53.8</td>
<td>55.5</td>
</tr>
<tr>
<td>-</td>
<td>Supervised</td>
<td>84.6</td>
<td>86.0</td>
</tr>
<tr>
<td>-</td>
<td>Semi-supervised</td>
<td>-</td>
<td>86.5</td>
</tr>
</tbody>
</table>

Compared to the previous work that targeted explicit relations [33] (with an F-measure of 42.7% versus 43.3%), our combined model achieved a significantly better F-measure, showing the contribution of implicit relation classifier.

### 6. Conclusions

In this work, we proposed an unsupervised method to bootstrap a relation classifier from knowledge graphs and query click logs. To mine natural language queries, we link the knowledge graph entities of interest to URLs using either the URL relations in knowledge graphs or the seed queries. Subsequently, we mine search queries from those URLs, label the queries with either explicit or implicit relations using different techniques, and then train relation classifiers with adaptive boosting. Apart from the classifier, we also get automatically labeled slots for explicit relations and query patterns for implicit relations as by-products. Our approach performs a significantly better F-measure (55.5%) than a natural baseline as well as previously published best results.

Upper bound and baseline. As a crude upper bound, we perform supervised learning on the labeled training data, which unsurprisingly gives a high F-measure of 86.0%.

As a simple baseline, we only output the most frequent relation (movie type), which gives an F-measure of 27.6%. Another baseline is the previously published best result on this dataset [33], using web search snippets that include two related entities to mine data for explicit relations. The main focus of that work is weighting knowledge graph entity types and enriching relation patterns by using word embeddings based on dependency parses. The F-measure with that approach is reported as 43.3%.

### Comparing URL sources.

Using URLs from knowledge graphs gives slightly better results than using URLs from seed queries. During error analysis, we discover that many seed queries q_t point to irrelevant URLs due to the ambiguity of the entity name. For example, the comic character Flash produces the seed query “flash movie,” which generally refers to Adobe Flash movies and not the comic character. In future work, we want to investigate the ways to obtain more accurate URLs from seed queries using, for example, the click statistics from QCL.

We will now discuss the results where the URLs come from knowledge graphs.

#### Single relation type.

The explicit relation classifier gives around 15% absolute increase in F-measure over the majority baseline. On the contrary, the implicit relation classifier gives relatively lower scores. This is mainly because there are fewer implicit relations than explicit ones, and the implicit relation classifier only covers 10 out of all 70 relations in the dataset.

#### Combined model.

The combined model achieves the F-measure of 55.5%, which is significantly higher than the score from the explicit relation classifier. This means both explicit and implicit relation datasets help boost the performance of the classifier.

While the explicit relation classifier trained with the search query logs data results in a similar F-measure to the previous work that targeted explicit relations [33] (with an F-measure of 42.7% versus 43.3%), our combined model achieved a significantly better F-measure, showing the contribution of implicit relation classifier.

#### Semi-supervised learning.

The bootstrapped classifier can also be used to improve the accuracy of the fully supervised model. By applying the best unsupervised classifier on the queries in the supervised learning dataset and use the predictions as additional features for semi-supervised learning, we are able to slightly increase the F-measure from 86.0% to 86.5%.
References


